Exploring and Exploiting Data-Free Model Stealing

Exploring and Exploiting Data-Free Model Stealing

Chi Hong[1], Jiyue Huang[1], Robert Birke[2], and Lydia Y. Chen[1] ✉

[1] Delft University of Technology, Delft, Netherlands
{c.hong, j.huang-4, Y.Chen-10}@tudelft.nl
[2] University of Torino, Turin, Italy
robert.birke@unito.it

Abstract. Deep machine learning models, e.g., image classifier, are increasingly deployed in the wild to provide services to users. Adversaries are shown capable of stealing the knowledge of these models by sending inference queries and then training substitute models based on query results. The availability and quality of adversarial query inputs are undoubtedly crucial in the stealing process. The recent prior art demonstrates the feasibility of replacing real data by exploring the synthetic adversarial queries, so called data-free attacks, under strong adversarial assumptions, i.e., the deployed classier returns not only class labels but also class probabilities. In this paper, we consider a general adversarial model and propose an effective data-free stealing algorithm, TandemGAN, which not only explores synthetic queries but also explicitly exploits the high quality ones. The core of TandemGAN is composed of (i) substitute model which imitates the target model through synthetic queries and their inferred labels; and (ii) a tandem generator consisting of two networks, $\mathcal{G}_x$ and $\mathcal{G}_e$, which first explores the synthetic data space via $\mathcal{G}_x$ and then exploits high-quality examples via $\mathcal{G}_e$ to maximize the knowledge transfer from the target to the substitute model. Our results on four datasets show that the accuracy of our trained substitute model ranges between 96–67% of the target model and outperforms the existing state-of-the-art data-free model stealing approach by up to 2.5X.

Keywords: model stealing · data-free · generative adversarial networks.

1 Introduction

Emerging intelligent services, such as Google translate and optical character recognition [8], are increasingly powered by deep models. Users can access these services by sending queries via APIs to get outputs, for instance, the class labels of queried images. While an open access to deployed models greatly eases users' experience, it opens up vulnerability issues related to model stealing [26, 29, 10]. Adversaries may use such an access to steal the knowledge of the deployed model and create a copy of it, named substitute model, which can then be used to do malicious actions, e.g., adversarial attacks [6, 14, 26, 27, 4] or membership

Authors' copy of C. Hong, J. Huang, R. Birke, and L. Y. Chen, "Exploring and Exploiting Data-Free Model Stealing," in European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Turin, Italy, 2023.

attacks [28]. Several studies design defenses against model stealing [21, 11, 12] , but [2] provides theoretical evidence that model extraction is inevitable.

To launch a model stealing attack, the adversary first needs to query the target model and get the corresponding inference results. For instance, to steal an image classifier, the adversary sends query images to the deployed classifier and then receives the predicted class labels. Recognizing the issue of limited availability of real data, recent studies [26, 10] introduce data-free model stealing methods, i.e., only using synthetic query images. Generative adversarial networks (GANs), composed of generator and substitute models, are key to synthesize queries. Synthetic query quality is paramount to extract knowledge from the target model. Low quality queries provide little information to train the substitute model, e.g. low confidence results from the target model leading to small feedback losses.

Though the existing studies demonstrate the feasibility to steal models in a (real) data-free way, they are limited in the adversarial assumptions and quality of synthesised queries. The common adversarial assumption of the prior art [26, 10] is that both predicted class labels and confidence are provided by the target model. The additional information beyond class labels is crucial for existing methods to carry out GANs training which can not backpropagate the gradients simply from the class labels.

Furthermore, the competition between generator and substitute models pushes the two networks to continuously **explore** the synthetic data space, but the classic minmax loss used in training GANs eschews to **exploit** synthetic examples [3]. As a result the average quality of synthetic queries is low, i.e., target model has low confidence in classifying synthetic data, and further limits the knowledge extraction performance in model stealing. Fig. 1 illustrates the average prediction confidence of the target model on real and synthetic data generated by the state of the art data-free stealing methods, namely DFME [26], DaST [29] and MAZE [10]. In this example, the target model is RestNet34 trained on CIFAR 10 and the settings of adversaries can be found in the evaluation section. Shown in the figure, the existing methods can only generate synthetic queries reaching an average confidence level of 80% that is 20% lower than the real data and limits its capacity to extract knowledge.

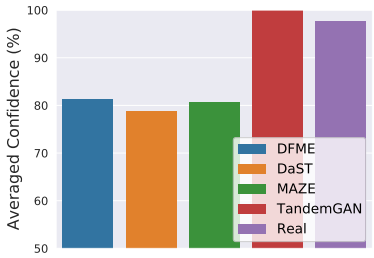


Fig. 1: Average target model confidence on 1K real data and 1K synthetic data generated by DFME, DaST, MAZE and TandemGAN respectively.

In this paper, we propose an effective data-free model stealing framework, TandemGAN, which does not require any real data and considers general adversarial assumptions where queries return label predictions, termed *label-only*, or label probabilities, termed *probability-only*. The core components of Tandem-

GAN are (i) the substitute model and (ii) the tandem generator consisting of two networks to explore and exploit the synthesizing space. The substitute model minimizes the distance loss between its and the target model's predictions while the two tandem generator networks jointly synthesize diverse and high confidence query examples which maximize the stealing potential. The first network, $\mathcal{G}_x$, explores the synthetic data space, whereas the second network $\mathcal{G}_e$ exploits and refines the synthesizing space to produce high-quality synthetic examples with high target model confidence. In the example of Fig. 4, we demonstrate more systematically the effectiveness of incorporating exploitation evaluating the performance of TandemGAN on stealing four deployed classifiers under both scenarios of label-only and probability-only against SOTA prior art.

The contributions of this paper are: (i) an effective data-free model stealing framework, TandemGAN, which uniquely features joint exploration and exploitation of synthetic data space and examples; (ii) more general adversarial scenarios: only class labels are available to the adversary; (iii) extensive evaluation and comparison against existing SOTA data-free model stealing approaches; and (iv) remarkable accuracy of the trained substitute models, i.e., reaching 67% up to 96% accuracy of the target classifiers.

2 Related work

Model stealing aims to distill the knowledge from a deployed (target) model, specifically, to train a highly similar substitute model [13, 2, 29, 26, 9, 20, 24]. A successful substitute model is able to obtain the implicit mapping function (or knowledge, at high level) of the target model via different (simpler) network structures [10, 20]. Two types of model stealing methods exist depending on whether the attackers have (partially) access to real training data or not. When real data is available, knowledge distilling [7, 19] extracts the knowledge of the target model through its class probabilities and transfers it to a lightweight substitute model. Without real data attackers can only query the target model through synthetic examples [18, 13, 29, 10] –a data-free approach.

The core of existing data-free model stealing methods [10, 26, 29] follows the design principle of GANs –competing generator-substitute networks. A generator produces synthetic examples to query the target model, $\mathcal{T}$, whereas the substitute model, $\mathcal{S}$, tries to imitate/steal $\mathcal{T}$ via the synthetic query results. The target model parameters and architecture are unknown. MAZE [10] and DFME [26] rely on a gradient approximation [22] to train their generator. DFME and MAZE can not be applied to scenarios where the target model provides only inference labels, Furthermore, DaST [29] regards the output of the target model as a constant vector forgoing the need for gradient approximation. Aforementioned studies explore the general synthesizing space, overlooking the option of exploitation. The features of different data-free model stealing methods are summarized in Tab. 1.

Table 1: Existing data-free model stealing methods.

Method	Probability-only	Label-only	Exploration	Exploitation
MAZE [10]	✓	✗	✓	✗
DFME [26]	✓	✗	✓	✗
DaST [29]	✓	✓	✓	✗
TandemGAN (ours)	✓	✓	✓	✓

3 Methodology

In this section, we introduce TandemGAN, a data-free model stealing framework that explores and exploits synthetic queries. Prior to introducing the design of TandemGAN, we first introduce the adversarial assumptions.

Adversarial assumptions. We consider a realistic deployment setting where a target classifier[3] $\mathcal{T}$ is deployed and its parameters and architecture of $\mathcal{T}$ are unknown. The only way to interact with the target model is by sending queries, e.g., images, and getting the inferred results, for both benign and malicious adversaries. Furthermore, due to the limitation and difficulty of obtaining real data, we further assume adversaries have no access to the real data. According to the format of the inference results, we consider two types of adversarial scenarios: (i) *label-only* scenario: $\mathcal{T}$ only provides a label prediction for each query without any additional information, and (ii) *probability-only* scenario: $\mathcal{T}$ returns the class probabilities instead.

3.1 TandemGAN framework

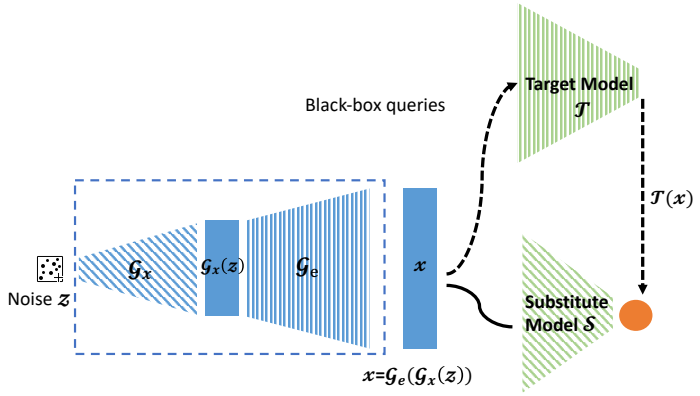


Fig. 2: TandemGAN framework: data-free model stealing process.

[3] TandemGAN can be extended to other model types but here we only discuss classification tasks.

We propose TandemGAN to steal the knowledge from $\mathcal{T}$ and train an accurate substitute model $\mathcal{S}$, shown in Fig 2. $\mathcal{S}$ can be regarded as a clone of $\mathcal{T}$ but with a different network architecture. Different from related work TandemGAN leverages a tandem generator which generates synthetic queries [4] with high classification confidence. Specifically, the generating process includes two networks, one network for exploring the new areas of the synthetic data space and one network for exploiting a particular space to generate high-quality synthetic queries. In Fig. 3, we illustrate the projected synthesizing space of TandemGAN on CIFAR10, the exploration points randomly scatter, whereas the exploitation points center around some exploration points. Note that we apply UMAP [16] to reduce the data dimension for visualization.

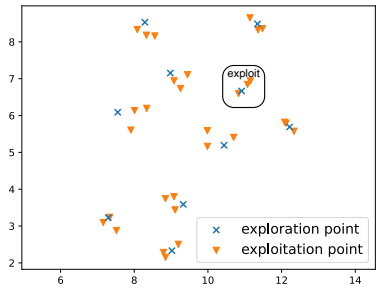


Fig. 3: Exploration and exploitation of TandemGAN in projected 2D space.

Preliminary. In the data-free model stealing, a generator $\mathcal{G}$ is fed a noise vector z as seed to generate a synthetic example $x = \mathcal{G}(z)$ to query $\mathcal{T}$ and obtain $\mathcal{T}(x)$. We assume $\mathcal{T}$ to be a model for classification task with N classes. Depending on the context, $\mathcal{T}(x)$ is either the predicted probability vector (*probability-only*) or predicted one-hot encoded label (*label-only*) for input x and $\mathcal{T}_i(x)$ denotes the i-th ($i = 1, ..., N$) class output. Similarly, we use $\mathcal{S}(x)$ to denote the probability output of the substitute model $\mathcal{S}$ for input x.

Algorithm architecture. Fig. 2 shows the architecture of our proposed model stealing framework, TandemGAN. The key components are the substitute model $\mathcal{S}$, and the tandem generator, $\mathcal{G}(z) = \mathcal{G}_e(\mathcal{G}_x(z))$, comprising $\mathcal{G}_x$ and $\mathcal{G}_e$. $\mathcal{G}$ generates synthetic queries that explore the synthetic data space via $\mathcal{G}_x$ and exploit a particular generation space via $\mathcal{G}_e$. We use $\theta_{\mathcal{S}}$, $\theta_{\mathcal{G}_x}$ and $\theta_{\mathcal{G}_e}$ to denote the model parameters of $\mathcal{S}$, $\mathcal{G}_x$ and $\mathcal{G}_e$, respectively.

During model stealing, a noise vector z is fed into $\mathcal{G}$ to produce a synthetic example x via $\mathcal{G}_x$ and $\mathcal{G}_e$. $\mathcal{G}_x$ transforms z into a latent code $\mathcal{G}_x(z)$ while $\mathcal{G}_e$ generates the query, e.g. image, from the latent code $x = \mathcal{G}_e(\mathcal{G}_x(z))$. x is used to query $\mathcal{T}$ and get the prediction $\mathcal{T}(x)$. Next, x associated with $\mathcal{T}(x)$ is used to train $\mathcal{S}$. When training $\mathcal{S}$, we minimize the distance measure to maximize the

[4] We refer the data sample sent to the target model as a query. In case of querying via synthetic data sample, we abbreviate it as a synthetic query.

agreement between $\mathcal{S}$ and $\mathcal{T}$. Besides training $\mathcal{S}$, we separately train $\mathcal{G}_x$ and $\mathcal{G}_e$ according to the outputs $\mathcal{T}(x)$ and $\mathcal{S}(x)$. Since $\mathcal{G}_x$ is designed to explore new areas of the data generating space which can provide new knowledge to train $\mathcal{S}$, we employ the design idea of GANs – a minmax optimization to train $\mathcal{G}_x$ and $\mathcal{S}$ jointly.

Hence to train $\mathcal{G}_x$, we optimize the model parameters $\theta_{\mathcal{G}_x}$ to maximize the distance measure between $\mathcal{T}(x)$ and $\mathcal{S}(x)$. $\mathcal{G}_e$ is responsible to refine the new area searched by $\mathcal{G}_x$ to generate high-quality synthetic queries. The objective of $\mathcal{G}_e$ thus needs to be aligned with the definition of query quality. Following the risk analysis of knowledge distillation [17], we define the optimization objective of $\mathcal{G}_e$ according to effectiveness of knowledge extraction, i.e., maximizing the confidence of the target model in predicting synthetic queries. Other possible definitions by active learning [1] are discussed in Sec. 5. As a result $\mathcal{G}$ simultaneously explores and exploits the synthetic data space to find diverse and highly informative examples which maximize the knowledge transfer to $\mathcal{S}$. The optimization objectives of $\mathcal{S}$, $\mathcal{G}_x$ and $\mathcal{G}_e$ are detailed in the following.

3.2 Optimization objectives and training procedure

Optimization objective of $\mathcal{S}$. Since $\mathcal{S}$ is a substitute of $\mathcal{T}$, their outputs are expected to be as similar as possible. Inspired by knowledge distillation [7], $\mathcal{S}$ imitates the outputs of $\mathcal{T}$ through the loss of distance measure. The loss function of $\mathcal{S}$ over a query example x is as follows:

$$\mathcal{L}_{\mathcal{S}} = \mathrm{D}(\mathcal{T}(x), \mathcal{S}(x)), \tag{1}$$

where $\mathrm{D}(\cdot)$ denotes the distance measure for loss, e.g., L1-Norm or cross-entropy. L1-Norm provides a stronger feedback than cross-entropy. This fits well the *probability-only* scenario where the loss inputs are the class probabilities. When working with the more limited scenario of *label-only*, the L1-Norm would require the output of the two models to be identical which is an aggressive condition given that only a one-hot output is provided rather than the full distribution on all classes. Thus, cross-entropy is applied for *label-only*. After training $\mathcal{S}$, we can steal the knowledge of $\mathcal{T}$ because $\mathcal{S}$ learns the mapping of $\mathcal{T}$.

Optimization objective of $\mathcal{G}_x$. We incorporate $\mathcal{G}_x$ to diversify the latent codes fed to $\mathcal{G}_e$ and generate queries in new areas of the data space. $\mathcal{G}_x$ aims at making $\mathcal{S}(x)$ as different as possible from $\mathcal{T}(x)$. This is the opposite training objective of $\mathcal{S}$. Thus, we formulate the loss of $\mathcal{G}_x$ as:

$$\mathcal{L}_{\mathcal{G}_x} = -\mathcal{L}_{\mathcal{S}} = -\mathrm{D}(\mathcal{T}(x), \mathcal{S}(x)), \quad \textit{where } x = \mathcal{G}_e(\mathcal{G}_x(z)). \tag{2}$$

By this means, $\mathcal{G}_x$ ensures that $\mathcal{S}$ is trained by a broad spectrum of diverse synthetic queries in data space to prevent model collapse or degenerated cases. It should be noted that although $\mathcal{T}(x)$ can not be differentiated directly via backpropagation (since its network parameters are unavailable), it is possible to apply gradient approximation (details below) under *probability-only* scenario. On

the other hand, under *label-only* scenario, the output of the target model is the class label and it is a non-differentiable constant.

Optimization objective of $\mathcal{G}_e$. We incorporate $\mathcal{G}_e$ to generate high quality queries around a latent space explored by $\mathcal{G}_x$. We derive the loss function of $\mathcal{G}_e$ inspired by the risk analysis of knowledge distillation [17]. We aim to achieve high quality $\mathcal{T}$ so as to better teach $\mathcal{S}$. Specifically, the quality of the probability estimate of $\mathcal{T}$ can be measured by log-loss and calibration error [5], the lower the better. Inspired by this observation, for minimizing the log-loss (or calibration error) on the outputs $\mathcal{T}(x)$, in our model stealing process, the inference confidence of x, i.e., the biggest element of $\mathcal{T}(x)$, is expected to be high. Consequently, the objective of $\mathcal{G}_e$ is to generate a synthetic query x that maximizes the confidence over model $\mathcal{T}$. We thus define high-quality queries as ones with high inference confidence on $\mathcal{T}$. Then, we define the loss function of $\mathcal{G}_e$ as:

$$\mathcal{L}_{\mathcal{G}_e} = -\left\{\log \mathcal{T}_k(x) \mid \forall_j : \mathcal{T}_j(x) \leq \mathcal{T}_k(x)\right\}, \qquad \text{where } x = \mathcal{G}_e(\mathcal{G}_x(z)). \tag{3}$$

With this loss, for an input example $x = \mathcal{G}_e(\mathcal{G}_x(z))$, we maximize the value of the k-th element of $\mathcal{T}(x)$ where k is the index of the biggest element. For *probability-only*, calculating Eq. (3) relies on gradient approximation (details below). For *label-only*, $\mathcal{T}(x)$ is a constant (one-hot label) and gradient approximation is not applicable because one can not obtain its directional derivative. Thus, we use $\mathcal{S}(x)$ to approximate $\mathcal{T}(x)$ since $\mathcal{S}(x)$ gradually approaches $\mathcal{T}(x)$ during training. Hence, updating $\mathcal{G}_e$ only needs the gradient of $\mathcal{S}$ which fits the *label-only* scenario.

Gradient approximation. Training the generator $\mathcal{G}$ requires the gradient $\nabla_{\theta_g}\mathcal{L}$ where $\mathcal{L}$ has two arguments $\mathcal{T}(x)$ and $\mathcal{S}(x)$. However, $\mathcal{T}$ is not differentiable in our black-box setting as its model parameters are unknown. Therefore we cannot obtain $\nabla_{\theta_g}\mathcal{L}$ without $\nabla_{\theta_g}\mathcal{T}(\mathcal{G}(z))$. To address this challenge, we apply gradient approximation [26] to approximate $\nabla_{\theta_g}\mathcal{L}$. Given

$$\nabla_{\theta_g}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \theta_g} = \frac{\partial \mathcal{L}}{\partial x} \times \frac{\partial x}{\partial \theta_g},$$

the second term can be computed because $\mathcal{G}$ is differentiable with known parameters. Thus, we only need to approximate $\frac{\partial \mathcal{L}}{\partial x}$. This can be done by the forward differences method [22]. It approximates the gradient of a function at a point by computing directional derivatives along some random directions. For a synthetic example $x \in \mathbb{R}^d$, the approximate gradient is:

$$\hat{\nabla}_x \mathcal{L}(x) = \frac{1}{M}\sum_{j=1}^{M} \frac{\mathcal{L}(x + \epsilon u_j) - \mathcal{L}(x)}{\epsilon} u_j$$

where u_j is a random direction (a d-dimensional unit vector) and M is the number of directions used for the approximation. ϵ is a small step size in the direction of u_j. The approximate value becomes more precise when M increases. For TandemGAN, gradient approximation is used to optimize both $\mathcal{G}_x$ and $\mathcal{G}_e$ under *probability-only* scenario.

Algorithm 1 TandemGAN

Input: n_x, n_e, n_s, μ_x, μ_e and μ_s
In the following, $z \sim \mathcal{N}(0,1)$
for *number of rounds* **do**
  for $i = 1, ..., n_x$ **do** `// explore the data space`
1 Get $\mathcal{T}(x)$ and $\mathcal{S}(x)$, $x = \mathcal{G}_e(\mathcal{G}_x(z))$
    if *probability-only* **then**
2 $\mathcal{L}_{\mathcal{G}_x}(x) = -\|\mathcal{T}(x) - \mathcal{S}(x)\|_1$
3 **else**
4 $\mathcal{L}_{\mathcal{G}_x}(x) = -CE(\mathcal{T}(x), \mathcal{S}(x))$
5 $\theta_{\mathcal{G}_x} \leftarrow \theta_{\mathcal{G}_x} - \mu_x \nabla_{\theta_{\mathcal{G}_x}} \mathcal{L}_{\mathcal{G}_x}(x)$
6 **for** $j = 1, ..., n_e$ **do** `// exploit the data space`
7 Get $\mathcal{T}(x)$ and $\mathcal{S}(x)$, $x = \mathcal{G}_e(\mathcal{G}_x(z))$
    Compute $\mathcal{L}_{\mathcal{G}_e}(x)$ `// see Eq. (3) for the loss calculation`
    $\theta_{\mathcal{G}_e} \leftarrow \theta_{\mathcal{G}_e} - \mu_e \nabla_{\theta_{\mathcal{G}_e}} \mathcal{L}_{\mathcal{G}_e}(x)$
8 **for** $j = 1, ..., n_s$ **do**
9 Get $\mathcal{T}(x)$ and $\mathcal{S}(x)$, $x = \mathcal{G}_e(\mathcal{G}_x(z))$
    if *probability-only* **then**
10 $\mathcal{L}_{\mathcal{S}}(x) = \|\mathcal{T}(x) - \mathcal{S}(x)\|_1$
11 **else**
12 $\mathcal{L}_{\mathcal{S}}(x) = CE(\mathcal{T}(x), \mathcal{S}(x))$
13 $\theta_{\mathcal{S}} \leftarrow \theta_{\mathcal{S}} - \mu_s \nabla_{\theta_{\mathcal{S}}} \mathcal{L}_{\mathcal{S}}(x)$
Result: The trained $\mathcal{S}$

Stealing algorithm Algorithm 1 shows the training process. In each round, there are two stages, exploration (line 1-5) and exploitation (line 6-7). In exploration, $\mathcal{G}$ and $\mathcal{S}$ are playing a min-max game just like GANs. By the game, $\mathcal{G}_x$ is updated to explore a new area in the latent space of $\mathcal{G}_e$. In exploitation, $\mathcal{G}_e$ exploits the area and produces synthetic examples to train $\mathcal{S}$. After the exploitation, we sample multiple examples for updating the substitute model $\mathcal{S}$ (line 8-13).

To fine-tune the balance between exploration and exploitation in each round, each training stage of the tandem generator is repeated n_x and n_e times, respectively. The optimization goal of $\mathcal{G}_x$ is opposite to $\mathcal{S}$. If the exploration is too aggressive, the training of $\mathcal{S}$ diverges. If the exploration is too conservative, $\mathcal{S}$ can collapse to a bad local optima because of the limited area searched in the latent space during training. After training $\mathcal{G}_e$, $\mathcal{S}$ needs enough updates to extract knowledge from $\mathcal{T}$.

4 Evaluation

In this section, we comprehensively evaluate the model stealing performance via the accuracy of the substitute model. We compare TandemGAN with state-of-the-art data-free model stealing approaches and conduct an ablation study to verify the effectiveness of exploration and exploitation, and the impact of different substitute model architectures.

Table 2: Comparison of the $\mathcal{S}$ubstitute model Acc(uracy) under DaST, MAZE, DFME, and TandemGAN. Arch. stands for model architecture.

	$\mathcal{T}$**arget**		$\mathcal{S}$	**Label-only**		**Probability-only**			
Dataset	Acc.	Arch.	Arch.	DaST	TandemGAN	MAZE	DFME	DaST	TandemGAN
MNIST	99.51	VGG16	VGG11	83.98	91.30	95.13	90.22	90.16	95.95
F-MNIST	93.09	VGG16	VGG11	43.00	72.15	41.63	48.43	44.43	79.96
SVHN	94.96	ResNet34	VGG11	58.39	62.41	52.60	50.62	55.69	68.80
CIFAR10	90.71	ResNet34	VGG11	21.28	29.58	58.67	54.79	29.81	75.81

Datasets and model structures. We evaluate our proposed method on four benchmark datasets: MNIST, Fashion-MNIST (F-MNIST), SVHN and CIFAR10. For MNIST and F-MNIST, we use VGG16 for the target model and ResNet34 for SVHN and CIFAR10. For the substitute model, we apply VGG11 for every dataset so that comparisons on the same/different network architecture(s) family between $\mathcal{S}$ and $\mathcal{T}$ are possible. However, for each baseline approach and TandemGAN, we use the same target and substitute models to guarantee a fair comparison. Finally, TandemGAN uses a two convolutional layers network for $\mathcal{G}_x$ and a one convolutional layer network for $\mathcal{G}_e$.

Evaluation criteria. The goal of our data-free model stealing is to achieve high classification **accuracy** on substitute model. We also compare the convergence process of TandemGAN with the baseline models to show the learning performance. For evaluating the attack efficiency, we also show the accuracy of the substitute model on different **number of queries** during model stealing.

Experiment settings. The networks $\mathcal{S}$, $\mathcal{G}_x$ and $\mathcal{G}_e$ are trained with a batch size of 256. We apply RMSprop as the optimizer for all the networks. The recommended learning rates for $\mathcal{S}$, $\mathcal{G}_x$ and $\mathcal{G}_e$ are 0.001, 10^{-5} and 10^{-6} respectively. $\mathcal{G}_x$ has two convolutional layers and $\mathcal{G}_e$ has one convolutional layer. We apply batch normalization after each convolutional layer on the generator. For gradient approximation, we set the number of random directions M to be 1 and choose the step size $\epsilon = 0.001$. To balance the training of $\mathcal{S}$, $\mathcal{G}_x$ and $\mathcal{G}_e$, we let $n_s = 5$, $n_x \in \{1, 3, 5\}$ and $n_e \in \{1, 3, 5\}$ for all experiments. We implement our method using pytorch. Intel Xeon E3-1200 CPUs and Nvidia GeForce RTX 2080 Ti GPUs are utilized to run the experiments. The code of our method will be released should the paper be accepted.

4.1 Model stealing performance

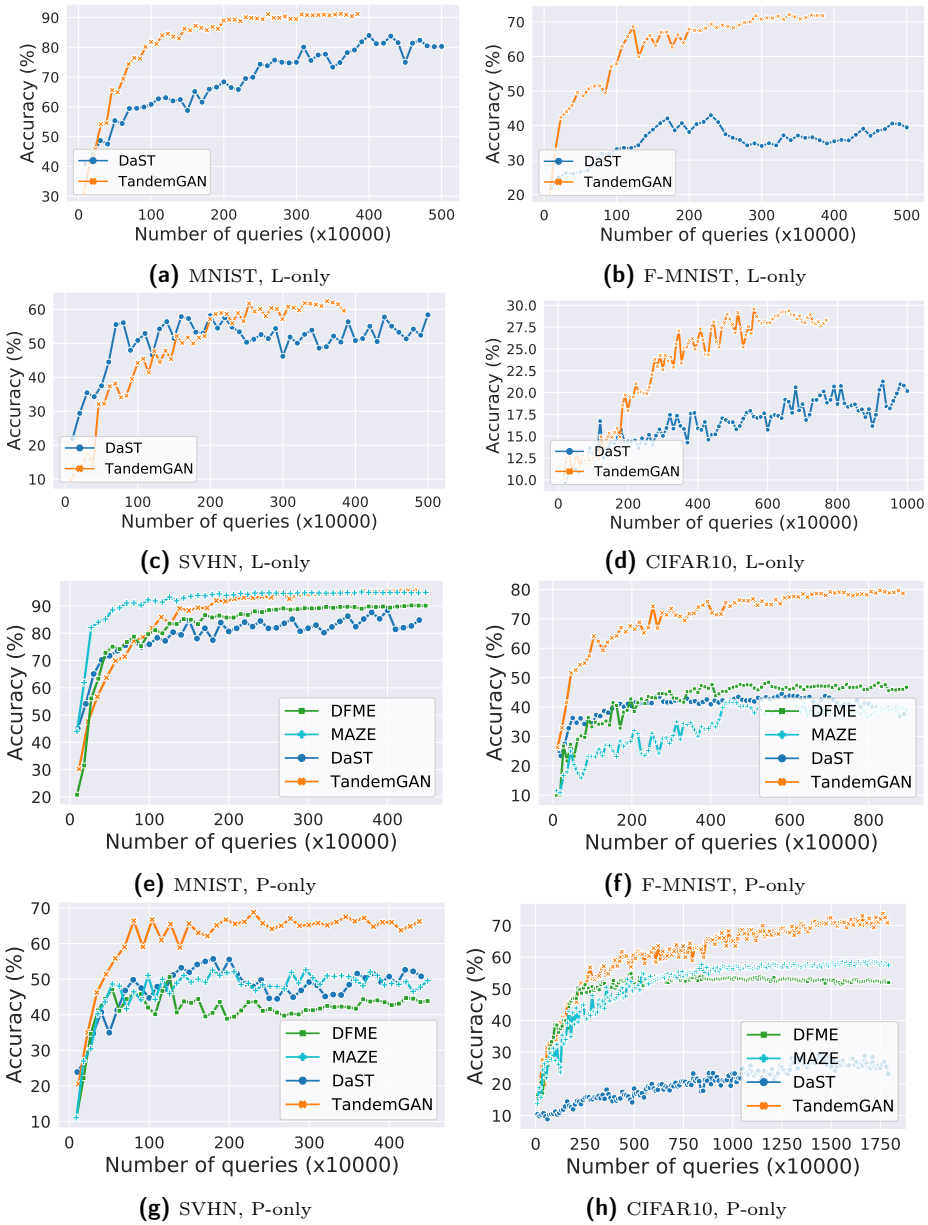
Model stealing accuracy. We evaluate the accuracy of model stealing results compared to the DaST, MAZE and DFME, for both *label-only* and *probability-only* scenarios, and the original target model. Tab. 2 and Fig. 4 summarize the results. Here we use VGG11 as substitute model. This is more challenging for SVHN and CIFAR10 since the target model is ResNet34, differing significantly in neural network architecture family. The accuracy of the target model represents the upper bound accuracy for the substitute model. It should be noted that

MAZE and DFME are only used on *probability-only* scenario since they require the additional information on class probabilities. We further note that DaST, MAZE, DFME and TandemGAN generate synthetic data per training iteration for generator or substitute model in different ways and different order. For a fair comparison, we report the accuracy under same number of queries.

Tab. 2 shows that TandemGAN achieves the highest accuracy among all datasets and inference scenarios. The accuracy of TandemGAN's substitute models trails behind the target models by margins of 4 to 32%, except for CIFAR10 under the label-only case. In other words, TandemGAN can achieve roughly 96 to 67% of the accuracy of the target model for a given dataset and inference scenario. The substitute model accuracy of TandemGAN is consistently and significantly higher than DaST, MAZE and DFME, showing an accuracy improvement up to 67% for the challenging label-only scenario, and up to 250% for the probability-only scenario.

Comparing probability-only to label-only, the accuracy of substitute models is higher for any given stealing method that is applicable to both scenarios. This is due to the fact that probability-only provides more inference information about the target model and we do not need to use $\mathcal{S}$ to approximate $\mathcal{T}$. For label-only scenarios, the accuracy of the substitute model trained by TandemGAN on MNIST surpasses 90% with a less than 10 percent points gap to the target model. This strongly demonstrates the effectiveness of TandemGAN. Owing to the increasing task difficulty, for F-MNIST, SVHN and CIFAR10, the stealing performance of TandemGAN drops. Even so the accuracy of the trained substitute models is still more than half of the target models (except CIFAR10), and 1.67x, 1.06x and 1.39x the accuracy achieved by DaST. For probability-only scenarios, the additional information provided by the class probabilities improves the accuracy of the substitute models. DFME outperforms MAZE and DaST on F-MNIST. MAZE outperforms DFME and DaST on MNIST and CIFAR10. DaST is better on SVHN. However, TandemGAN outperforms all. More impressive, TandemGAN achieves results close to the target model. The gap is limited from 3 to 26 percent points.

Model stealing convergence. Fig. 4 shows the evolution of the substitute model accuracy across the number of queries to the target model. For DaST, we can see that for all cases the accuracy fluctuates and does not converge at a good local optima during the entire training.Sometimes the accuracy does not even show an increasing trend, for instance Fig. 4b, 4f and 4h. Due to the unstable convergence, another issue with DaST is to choose an appropriate stopping criteria for training. Further, DaST saves the substitute model at each iteration, and chooses the one with the highest accuracy to be the final result. We argue that this is normally infeasible because attackers do not have real data to evaluate their saved substitute models. It also requires additional resources and efforts to store and select the best substitute model once the model stealing process ends. This further leads to the unstable training process of DaST. For MAZE and DFME, the convergence has no significant oscillation. The convergence of TandemGAN is almost monotonic. The accuracy increases smoothly during the



(a) MNIST, L-only **(b)** F-MNIST, L-only **(c)** SVHN, L-only **(d)** CIFAR10, L-only **(e)** MNIST, P-only **(f)** F-MNIST, P-only **(g)** SVHN, P-only **(h)** CIFAR10, P-only

Fig. 4: The convergence of accuracy of substitute models during training.

whole training, and converges to local optima at around 2M queries for MNIST and SVHN, and 4M for F-MNIST respectively. For CIFAR10, label-only scenario requires 6M queries while it takes around 13M for probability-only scenario. The reason behind the different number of queries is that label-only scenario provides limited information which cannot improve the accuracy even with increased queries for this relatively difficult task. This observation also implies that less information provided by the target model may serve as the short board for challenging tasks to apply model stealing attack.

Inherited from GANs, the generator design of DaST, MAZE and DFME only contains exploration to train the generator and substitute model in a min-max game. In TandemGAN, besides $\mathcal{G}_x$ to search for new space areas which generate diverse data examples, the tandem generator also contains $\mathcal{G}_e$ to exploit the latent code generated by $\mathcal{G}_x$, fine-tuning the task-specific data properties to train $\mathcal{S}$ for model stealing. Benefiting from both exploration and exploitation, our synthetic data captures the real data training scenarios better, resulting in stable convergence towards a good local optima.

4.2 Ablation study

Table 3: Ablation study for exploration and exploitation.

Ablation	TandemGAN
main results	75.81
w/o exploration	60.74
w/o exploitation	67.23

Table 4: Analysis for different substitute model architectures.

Architecture	TandemGAN
VGG11	75.81
ResNet18	84.70
AlexNet	23.25

Importance of both exploring and exploiting. In previous sections we highlight the advantages to additionally exploit, and not only explore, the synthetic examples. To quantify the benefits of either phase, we present an ablation study using VGG11 as $\mathcal{S}$ for CIFAR10 where TandemGAN forgoes either $\mathcal{G}_x$ (w/o exploration) or $\mathcal{G}_e$ (w/o exploitation). This is achieved by skipping the corresponding training phase, i.e. keeping either $\theta_{\mathcal{G}_x}$ or $\theta_{\mathcal{G}_e}$ fixed. Tab. 3 shows the results. One clearly sees that both exploring the synthetic data space and exploiting known examples holds the best results. Without exploitation the accuracy drops by 8 percent. This is in line with the result obtained by DFME which also performs only exploration (see Tab 2). Without exploration the accuracy is reduced by 15 percent. This shows that exploration and exploitation are both important for training a high accuracy substitute model.

Impact of architecture choice. As the neural network architecture is unknown by the attacker, we evaluate the impact of choosing different architectures for $\mathcal{S}$ Besides VGG11, we try ResNet-18, which is of the same neural network family of the target model (using ResNet34), as well as AlexNet which is a simpler CNN [15, 25]. The accuracy of the substitute AlexNet is low, i.e., 23.25% (see Tab 4). The reason is that AlexNet contains 5 convolutional layers and 3

Table 5: Comparison of the $\mathcal{S}$ubstitute model Acc(uracy) under DaST, MAZE, DFME, and TandemGAN. Arch. stands for model architecture.

	$\mathcal{T}$**arget**		$\mathcal{S}$	**Label-only**		**Probability-only**	
Dataset	Acc.	Arch.	Arch.	Public Samples	TandemGAN	Public Samples	TandemGAN
F-MNIST	93.09	VGG16	VGG11	37.08	72.15	32.06	79.96
CIFAR10	90.71	ResNet34	VGG11	15.45	29.58	14.01	75.81

fully connected layers which is too shallow to train on CIFAR10. Choosing a suitable task-specific substitute model architecture is crucial.ResNet18 achieves the highest accuracy. While ResNet18 is able to achieve slightly better accuracy than VGG11 on CIFAR10, i.e. 92.36% against 91.6%, we impute the performance gap of 9 percent points mainly to the sharing of the same neural network family between $\mathcal{S}$ and $\mathcal{T}$. Since the exact architecture of $\mathcal{T}$ is unknown to attackers we choose VGG11 for our main results. Using VGG11 we can better verify broadly the effectiveness and generality of TandemGAN.

Stealing the target using publicly available data samples. It is possible to query a target model and steal it using publicly available data samples (E.g., MNIST samples etc), when the real training dataset of the target model is unknown. To study the effectiveness of directly using publicly available data samples for model stealing, in the following, we apply MNIST samples to steal a F-MNIST target model and use SVHN samples to steal a CIFAR10 target model. The results are shown in Tab. 5. We can see that TandemGAN significantly outperforms the baseline which directly uses public samples for querying. Besides, comparing to Tab. 2, DaST, MAZE and DFME also outperform the baseline. Directly using public data samples to query cannot effectively and efficiently search the input data space of the target model, especially when the model and the task are complicated e.g., the CIFAR10 target. It also cannot utilize the inference feedback from the target to adjust the stealing process. That's why it is much worse than DaST, MAZE, DFME and TandemGAN which apply data sample generators for data space searching. In a data-free scenario, it is important to design a good data space searching strategy to perform model stealing attacks.

5 Possible extension

In this section, we discuss the possible extension of the proposed algorithm. In the following, we show some other perspectives of exploitation.

In the methodology part, we derive the optimization objective of $\mathcal{G}_e$ from a statistical observation [17] that the quality of a target (or a teacher) $\mathcal{T}$ can be measured by log-loss and calibration error (the lower the better). Since we regard the predictions from $\mathcal{T}$ as the ground truth labels in model stealing, in order to minimize the log-loss (or calibration error) on $\mathcal{T}$ we aim to increase the inference confidence of each data example x. Then we define the loss (see Eq. (3)) according to this motivation. From the statistical perspective, the high-quality

examples produced by exploitation are defined as examples with high inference confidence on $\mathcal{T}$. Actually, it is possible to extend TandemGAN by designing the optimization objective of $\mathcal{G}_e$ from other perspectives.

From active learning [23] perspectives, $\mathcal{T}$ can be seen as an oracle or a expert who can provide ground truth labels for unlabeled examples. In order to train $\mathcal{S}$, we need examples to query $\mathcal{T}$ and get the predictions. For efficiently and effectively querying $\mathcal{T}$, we need query strategies to select informative examples (high-quality examples). Therefore the query strategies, e.g., variance reduction, entropy sampling and margin sampling etc, of active learning can be utilized to define high-quality examples and design the optimization objective of $\mathcal{G}_e$.

If attackers know some prior information about the training data space of $\mathcal{T}$, e.g., data distribution, we can also consider the information when designing the loss of $\mathcal{G}_e$. In this case, $\mathcal{G}_e$ can be utilized to ensure that the data examples are sampled from the prior distribution. A special case is to consider the class balance of the generated examples when training $\mathcal{G}_e$.

6 Conclusion

It is challenging to design adversarial attacks without knowing the target model parameters nor having access to real-world data. In this paper, we propose a novel and effective data-free model stealing framework, TandemGAN, consisting of substitute model and a tandem generator networks, which aims to steal the knowledge of the target model by synthetic queries. Beyond the state of the art, we not only consider a general adversarial scenario with only the availability of predicted class labels only but also design a steal optimization algorithm to explore and exploit synthetic queries generation. We empirically demonstrate that TandemGAN effectively steals the target model using a small number of queries for four datasets. Under various adversarial scenarios, we show that the model stolen through TandemGAN achieves up to 2.5 times higher accuracy than state-of-the-art data-free model stealing attacks, and its accuracy is as high as the $96 - 67\%$ of target model.

Acknowledgements This work has been supported by the Spoke "FutureHPC & BigData" of the ICSC–Centro Nazionale di Ricerca in "High Performance Computing, Big Data and Quantum Computing", funded by European Union – NextGenerationEU and the EuPilot project funded by EuroHPC JU under G.A. n. 101034126.

References

1. Aggarwal, C.C., Kong, X., Gu, Q., Han, J., Yu, P.S.: Active learning: A survey pp. 571–606 (2014)
2. Chandrasekaran, V., Chaudhuri, K., Giacomelli, I., Jha, S., Yan, S.: Exploring connections between active learning and model extraction. In: USENIX Security. pp. 1309–1326 (2020)

3. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative Adversarial Nets. In: NIPS (2014)
4. Granese, F., Picot, M., Romanelli, M., Messina, F., Piantanida, P.: Mead: A multi-armed approach for evaluation of adversarial examples detectors. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022. pp. 286–303. Springer (2023)
5. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. PMLR (2017)
6. Gupta, V., Chakraborty, T.: Viking: Adversarial attack on network embeddings via supervised network poisoning. In: PAKDD. pp. 103–115. Springer (2021)
7. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. CoRR **abs/1503.02531** (2015)
8. Huang, Y., Tu, S., Ahmad, B., et al.: Learning a semantic space for modeling images, tags and feelings in cross-media search. In: PAKDD. pp. 65–76 (2019)
9. Juuti, M., Szyller, S., Marchal, S., Asokan, N.: PRADA: protecting against DNN model stealing attacks. In: IEEE EuroS&P. pp. 512–527 (2019)
10. Kariyappa, S., Prakash, A., Qureshi, M.K.: MAZE: data-free model stealing attack using zeroth-order gradient estimation. In: IEEE/CVF CVPR (2021)
11. Kariyappa, S., Prakash, A., Qureshi, M.K.: Protecting DNNs from theft using an ensemble of diverse models. In: ICLR (2021)
12. Kariyappa, S., Qureshi, M.K.: Defending against model stealing attacks with adaptive misinformation. In: 2020 IEEE/CVF CVPR. pp. 767–775 (2020)
13. Krishna, K., Tomar, G.S., Parikh, A.P., Papernot, N., Iyyer, M.: Thieves on sesame street! model extraction of BERT-based APIs. In: ICLR (2020)
14. Lin, Z., Shi, Y., Xue, Z.: Idsgan: Generative adversarial networks for attack generation against intrusion detection. In: PAKDD. pp. 79–91. Springer (2022)
15. Lu, S., Lu, Z., Zhang, Y.D.: Pathological brain detection based on alexnet and transfer learning. Journal of computational science **30**, 41–47 (2019)
16. McInnes, L., Healy, J.: UMAP: uniform manifold approximation and projection for dimension reduction. CoRR **abs/1802.03426** (2018)
17. Menon, A.K., Rawat, A.S., Reddi, S.J., Kim, S., Kumar, S.: A statistical perspective on distillation. In: ICML. vol. 139, pp. 7632–7642 (2021)
18. Micaelli, P., Storkey, A.J.: Zero-shot knowledge transfer via adversarial belief matching. In: NeurIPS. pp. 9547–9557 (2019)
19. Nguyen, D., Gupta, S., Nguyen, T., Rana, S., Nguyen, P., Tran, T., Le, K., Ryan, S., Venkatesh, S.: Knowledge distillation with distribution mismatch. In: Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021. pp. 250–265. Springer (2021)
20. Orekondy, T., Schiele, B., Fritz, M.: Knockoff nets: Stealing functionality of black-box models. In: IEEE CVPR. pp. 4954–4963 (2019)
21. Orekondy, T., Schiele, B., Fritz, M.: Prediction poisoning: Towards defenses against DNN model stealing attacks. In: ICLR (2020)
22. Polyak, B.T.: Introduction to optimization. optimization software. Inc., Publications Division, New York **1** (1987)
23. Ren, P., Xiao, Y., Chang, X., Huang, P.Y., Li, Z., Gupta, B.B., Chen, X., Wang, X.: A survey of deep active learning. ACM Comput. Surv. (CSUR) **54**(9), 1–40 (2021)
24. Sanyal, S., Addepalli, S., Babu, R.V.: Towards data-free model stealing in a hard label setting. In: IEEE/CVF CVPR. pp. 15284–15293 (2022)
25. Şeker, A.: Evaluation of fabric defect detection based on transfer learning with pre-trained AlexNet. In: IDAP. pp. 1–4. IEEE (2018)

26. Truong, J., Maini, P., Walls, R.J., Papernot, N.: Data-free model extraction. In: IEEE/CVF CVPR 2021. pp. 4771–4780 (2021)
27. Vijayaraghavan, P., Roy, D.: Generating black-box adversarial examples for text classifiers using a deep reinforced model. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Proceedings, Part II. pp. 711–726. Springer (2020)
28. Yeom, S., Giacomelli, I., Fredrikson, M., Jha, S.: Privacy risk in machine learning: Analyzing the connection to overfitting. In: IEEE CSF. pp. 268–282 (2018)
29. Zhou, M., Wu, J., Liu, Y., Liu, S., Zhu, C.: Dast: Data-free substitute training for adversarial attacks. In: 2020 IEEE/CVF CVPR. pp. 231–240 (2020)